26-Polymorphism

Polymorphism in Java

Definition: Polymorphism in Java is the ability of an object to take on many forms. The term "polymorphism" is derived from two Greek words***: poly***, meaning "**many**," and ***morphism***, meaning "**forms**" or "**behaviors**." Polymorphism allows objects to be treated as instances of their parent class while still allowing them to execute methods in a child-specific way.

Types of Polymorphism:

1. **Compile-time polymorphism (static polymorphism):**
   - In this type, the method behavior is determined at compile-time. This is achieved through **method overloading**.
2. **Runtime Polymorphism (Dynamic Polymorphism):**
   - In this type, the method behavior is determined at runtime. This is achieved through **method overriding**.

Explanation of Polymorphism Types:

- **Compile-time Polymorphism:**
  - Compile-time polymorphism is also known as **static polymorphism** or **early binding.** In this type of polymorphism, the method to be called is determined at the time of compilation based on the method signature.
  - **Method overloading** is a common example of compile-time polymorphism. It allows multiple methods to have the same name but differ by the number or type of parameters.

Example of Method Overloading:

```
public class Calculator {
    // Overloaded method with two parameters
    public int add(int a, int b) {
        return a + b;
    }

    // Overloaded method with three parameters
    public int add(int a, int b, int c) {
        return a + b + c;
    }
}
```

- **Runtime Polymorphism:**
    - Runtime polymorphism is also known as **Dynamic Polymorphism** or **Late Binding**. In this type, the method to be executed is determined at runtime, depending on the object's actual class.
    - **Method Overriding** is a common example of runtime polymorphism. It allows a subclass to provide a specific implementation of a method that is already defined in its parent class.

Example of Method Overriding:

```
class Animal {
    // Method in the parent class
    public void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    // Overriding the sound method in the subclass
    @Override
    public void sound() {
        System.out.println("Dog barks");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal myDog = new Dog(); // Runtime Polymorphism
        myDog.sound(); // Outputs: Dog barks
    }
}
```

Comparison of Method Overloading and Method Overriding:

- **Method Overloading:**
    - Occurs within the same class.

    - Involves methods with the same name but different parameters.
    - It is a compile-time concept.
- **Method Overriding:**
    - Occurs between a superclass and a subclass.
    - Involves methods with the same name, parameters, and return type.
    - It is a runtime concept.

Advantages of Polymorphism:

- **Code Reusability:** Polymorphism allows you to reuse existing code more efficiently.
- **Flexibility:** You can write more flexible and maintainable code.
- **Simplified Interface:** Different types of objects can be accessed through the same interface, simplifying code interactions.